CRYPTEXT CORPORATION

P.O. BOX 425
NORTHGATE STATION
SEATTLE, WA 98125
(206) 364-8585

PRICE LIST
Effective November 30, 1979

CRYPTEXT UNIT: includes--power cable --extension cable F-M-M --DEMO Software --USERS MANUAL	$ 389.00

CODEFILE Software

A disk-based file encryption program, supplied on DOS or formatted disk.

--on DOS disk (required for single disk drive systems)	$ 44.95
--on formatted disk	$ 29.95

DEMONSTRATION Software (supplied with CRYPTEXT)

An assembly language program allowing tape storage of encrypted information. Supplied on tape.

USERS MANUAL (supplied with CRYPTEXT) $ 4.95

The price of the manual is credited to the cost of a subsequent purchase of a CRYPTEXT.

EXTENSION CABLE (supplied with CRYPTEXT) $ 29.95

Due to possible strain on the computer PC Board, the manufacturer recommends the use of an extension cable.

Additional Power Cable $ 9.95

Apple interface available in December
price should be about $125.00
Chan

CRYPTEXT CORPORATION
P.O. BOX 425
NORTHGATE STATION
SEATTLE, WA 98125
(206) 364-8585

DEALER PRICE SCHEDULE
Effective November 30, 1979

2--9 units	25% off list
10 or more	30% off list

PRICE LIST
Effective November 30, 1979

CRYPTEXT UNIT: $ 389.00
includes--power cable
--extension cable F-M-M
--DEMO Software
--USERS MANUAL

CODEFILE Software
A disk-based file encryption program, supplied on DOS or formatted disk.
--on DOS disk (required for single disk drive systems) $ 44.95
--on formatted disk $ 29.95

DEMONSTRATION Software (supplied with CRYPTEXT)
An assembly language program allowing tape storage of encrypted information. Supplied on tape.

USERS MANUAL (supplied with CRYPTEXT) $ 4.95
The price of the manual is credited to the cost of a subsequent purchase of a CRYPTEXT.

EXTENSION CABLE (supplied with CRYPTEXT) $ 29.95
Due to possible strain on the computer PC Board, the manufacturer recommends the use of an extension cable.

Additional Power Cable $ 9.95

CRYPTEXT
CORPORATION
P.O. BOX 425
NORTHGATE STATION
SEATTLE, WA. 98125
(206) 364-8585

INTRODUCTORY
PRICE LIST
July 30, 1979

CRYPTEXT UNIT: includes--power cable --DEMO software	$ 299.00

EXTENSION CABLE (due to possible strain on the computer PC Board, the manufacturer recommends the use of an EXTENSION CABLE)

Female to male	$ 17.95
Female to male to male	$ 21.95

DEMONSTRATION Software (free w/CRYPTEXT)
An assembly language program allowing tape storage of encrypted messages. Supplied on tape.

CODEFILE Software
A disk-based file encryption program, supplied on DOS or formatted disk.

--on DOS disk (required for single disk systems)	$ 44.95
--on formatted disk	$ 29.95

Additional Power Cable	$ 9.95

Prices subject to change without notice.

CRYPTEXT

THE DAWN OF SECURITY

CRYPTEXT

USERS MANUAL

CRYPTEXT CORPORATION

P.O. BOX 425
NORTHGATE STATION
SEATTLE, WA. 98125
(206) 364-8585

LIMITED WARRANTY

Units deemed by the manufacturer to have been defective or inoperative from the date of manufacture will be replaced for a period of 90 days from date of manufacture. In implementing this replacement policy, decisions made by the manufacturer shall be deemed final and conclusive. No other warranty is made or offered, express or implied, including warranties of merchantability or suitability for any intended use.

Since CRYPTEXT Corporation has no control over the intended or actual use of its cryptographic products, it hereby specifically disclaims any responsibility or liability of any kind, direct, contingent or consequential, for any reason, for the use or misuse of any of its products or techniques or for any errors of commission or omission in information or documentation supplied by it.

All products, services and information supplied by CRYPTEXT Corporation are believed to be correct and accurate at the time they are supplied. CRYPTEXT Corporation reserves the sole right to modify its product designs and to change, augment or rescind, without notice, any or all of its services, techniques and documentation at any time.

CORPORATION

P.O. BOX 425
NORTHGATE STATION
SEATTLE, WA. 98125
(206) 364-8585

CRYPTEXT

USERS MANUAL

Table of Contents

the Quick & Dirty--USE IT--guide

If you just can't stand to read the instructions:

1. TURN YOUR ENTIRE COMPUTER OFF
2. ATTACH CRYPTEXT AS SHOWN IN FIGS. 1 & 2
3. ATTACH THE POWER CORD AS SHOWN IN THE FIGURES
4. Turn your computer on
5. Load the <u>system</u> DEMO program
6. Go for it!!!!!

Attaching CRYPTEXT to your TRS-80*

DAMAGE may occur to the electronics of the CRYPTEXT if the computer is not turned off or unplugged .

So first and always, turn off or unplug your entire computer before attaching your CRYPTEXT.

The CRYPTEXT unit easily plugs into the Radio Shack TRS-80 computer, either in the port on the left rear of the keyboard/CPU (as shown in figure 1) or in the screen printer port of the Expansion Interface (as shown in figure 2).

To attach the CRYPTEXT directly to the keyboard, simply unsnap the small plastic cover and plug the CRYPTEXT unit, lettered side up, onto the connector which is just visible inside. (Figure 1 shows this connection using the optional extension cable.)

CAUTION: Although the CRYPTEXT can be plugged directly onto the circuit boards in the Keyboard/CPU or Expansion Interface, downward pressure on the CRYPTEXT could cause damage to the circuit boards. The manufacturer strongly recommends the use of an extension cable to prevent damage.

To attach the CRYPTEXT to the Expansion Interface, identify the screen printer port, and detach the plastic cover there. Plug the CRYPTEXT onto the exposed connector (figure 2 shows this connection made using the optional extension cable).

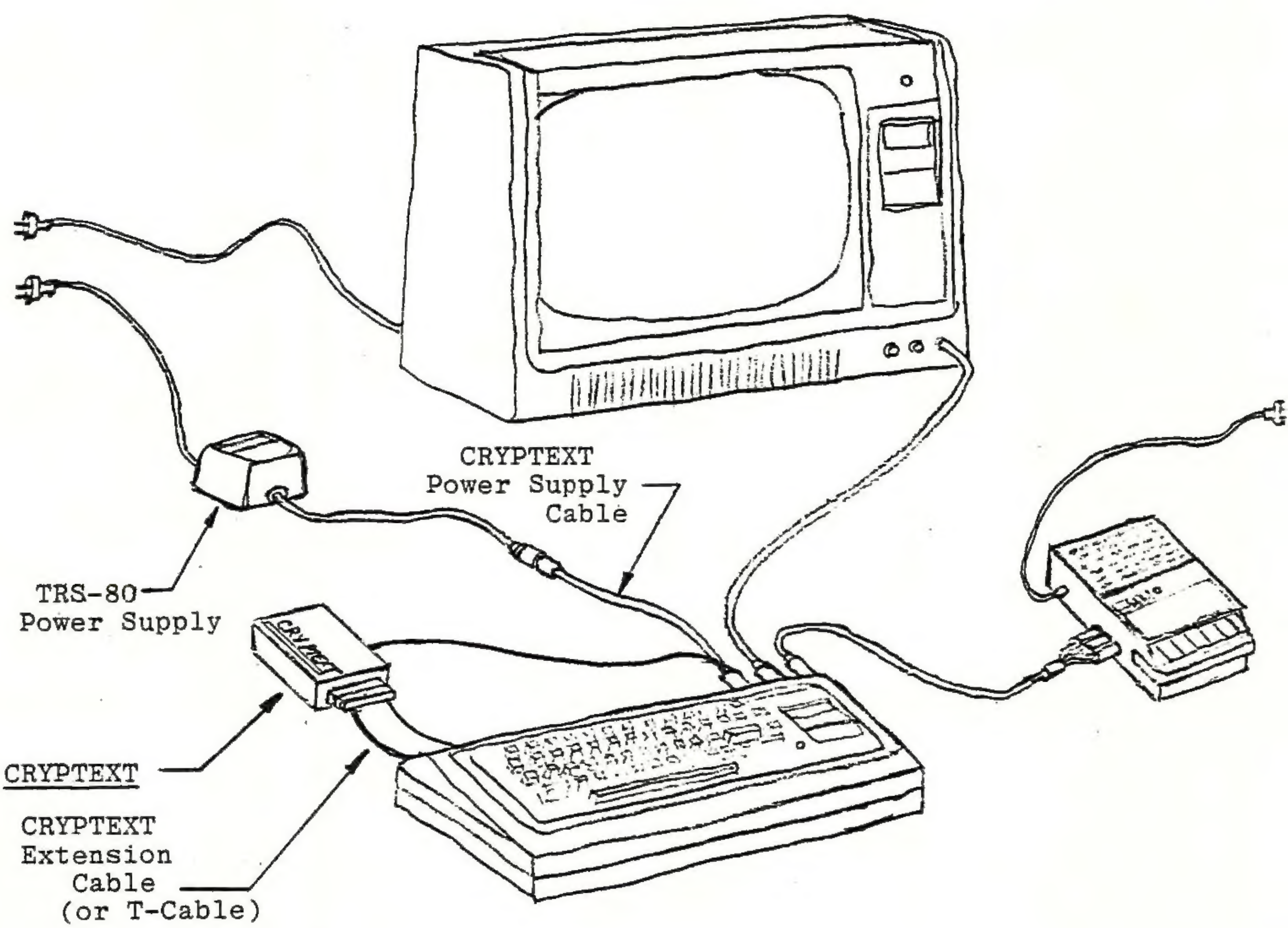
Attach the power cable supplied with the CRYPTEXT unit as shown in either figure 1 or 2. The power cable has different terminations at each end, so it can only be connected in the correct fashion. LOOK at the cable ends before plugging them in: don't use unnecessary force.

The computer can now be plugged in and turned on.

* TRS-80 is a trademark of TANDY Corporation.

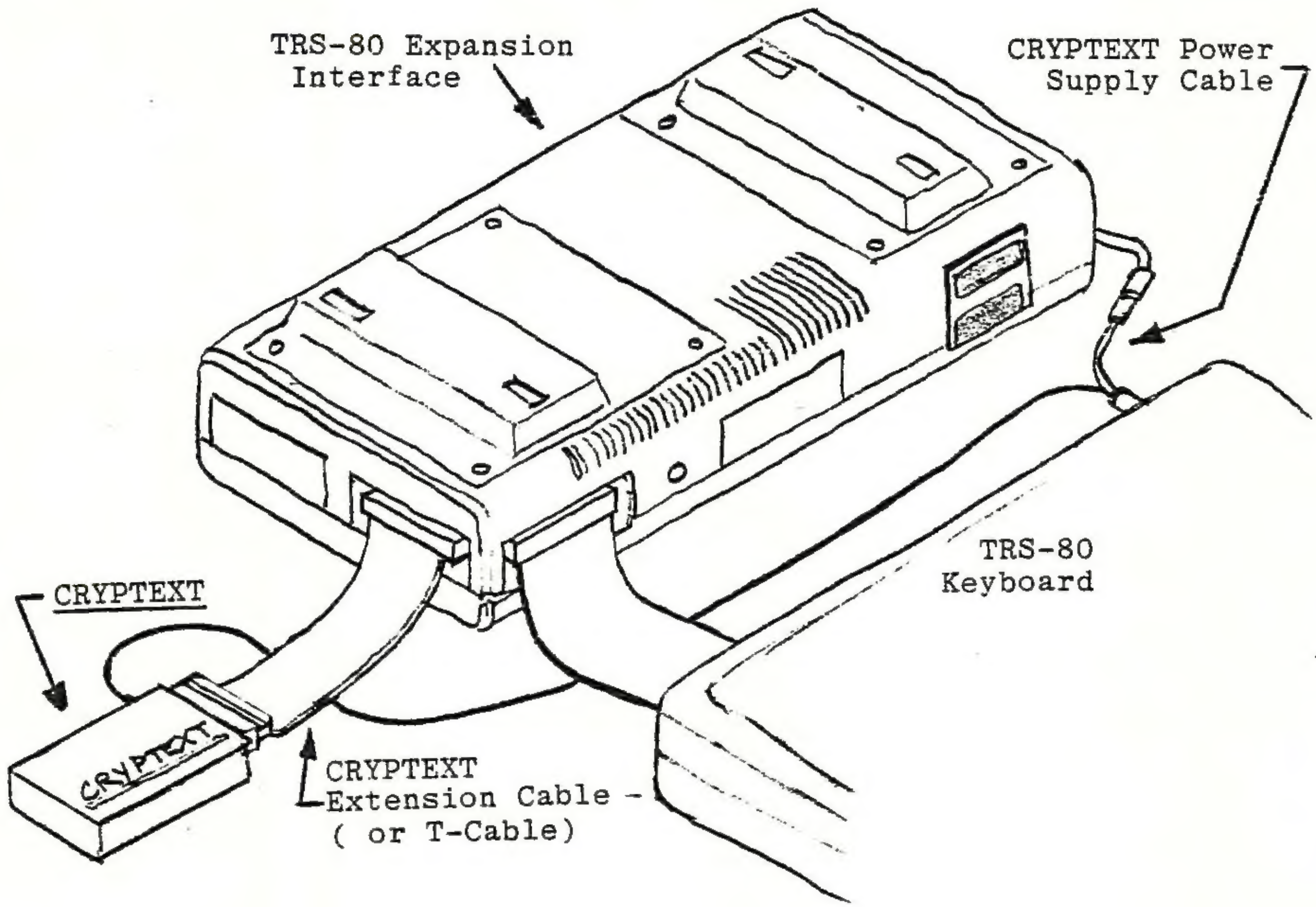
Attaching CRYPTEXT to your TRS-80

Figure 1



Attaching CRYPTEXT to your TRS-80

Figure 2



The DEMONSTRATION Program

Loading DEMO

The DEMONSTRATION Program supplied with your CRYPTEXT is an assembly language tape, and so is loaded using the normal system procedure.

After your computer is turned on:

TRS-80 will display:	your response should be:		
MEMORY SIZE?_	press (ENTER)		
RADIO SHACK LEVEL II BASIC READY >_	type SYSTEM	press	(ENTER)
	ready cassette recorder and play		
*?_	type DEMO	press	(ENTER)
	loading the tape takes about 55 seconds		
*?_	type /	press	(ENTER)

The DEMONSTRATION Program should be operating now. Respond to questions as they are asked. The program will, in the normal course of operation, provide enough information for you to respond properly.

<u>In case of difficulty</u>, or to end the demonstration, just press the BREAK key. The TRS-80 will return to Level II Basic.

During loading, two asterisks should appear in the upper right-hand corner of the screen. This is normal. If they do not appear or only one appears, or if a C appears, turn your computer off, then on again, and begin the loading procedure again, trying a different volume level on your recorder.

The DEMONSTRATION Program

Exercising CRYPTEXT with DEMO

With the DEMONSTRATION Program supplied with the CRYPTEXT unit, messages and files typed at the TRS-80 keyboard can be securely encoded and recorded on tape cassettes.

The reverse side (side 2) of the DEMONSTRATION Program cassette has been left blank and can be used for message storage if desired. (Any standard tape cassette can be used: however, if you use a tape which has a leader , be sure to advance the tape past the leader before attempting to record.)

The KEYWORD which you are asked to supply can be any combination of letters, numbers, punctuation, or other characters, lower or upper case (lower and upper case letters appear the same on the screen but are different), up to 10 characters long. (DEMO truncates on the right if you supply too many characters, and fills in with spaces if you choose to use fewer than 10 characters.)

Be sure to remember this KEYWORD!!!!! Your message will be lost if you forget the exact keyword. Remember: this is a secure storage system! The displays created by using the wrong keyword can be quite interesting.

Writing Programs in BASIC

to use the CRYPTEXT

FUNDAMENTALS

You will find it easy and convenient to write BASIC programs to have the TRS-80 encrypt information for any purpose you choose. Four steps must be accomplished by a BASIC program to use the CRYPTEXT:

1. Supply CRYPTEXT with a 1Ø-byte key.
2. Have CRYPTEXT initialize its code generators.
3. Supply a byte to CRYPTEXT to be encoded or decoded.
4. Obtain the corresponding encoded or decoded byte from the CRYPTEXT.

Exchanging information between the computer and the CRYPTEXT is accomplished by using the BASIC <u>OUT port, value</u> and <u>INP (port)</u> functions. (See the Radio Shack manual for more information on these two commands.)

EXAMPLE PROGRAMMING

The following subroutine asks for a 1Ø-byte key, K$, from the keyboard, initializes the CRYPTEXT, encodes a previously-stored string, S$, and assigns the encoded string the name B$.

```
1Ø1Ø DEFINT I:  B$=""
1Ø2Ø INPUT "ENTER 1Ø CHARACTERS";K$: IF LEN(K$)><1Ø
     THEN 1Ø2Ø
1Ø3Ø FOR I=1 TO 1Ø: OUT 194,ASC(MID$(K$,I,1)):
     OUT 196+I,Ø: NEXT I: K$=""
1Ø4Ø OUT 195,255
1Ø5Ø IF INP (193)=255 THEN 1Ø5Ø
1Ø6Ø FOR I=1 TO LEN(S$): OUT 194,ASC(MID$(S$,I,1)):
     B$=B$+CHR$(INP(193)): NEXT I: RETURN
```

INSTRUCTION DETAILS

Line 1Ø3Ø

Line 1Ø3Ø issues the key to the CRYPTEXT, one byte at a time.

The statement OUT 194,ASC(MID$(K$,I,1)) sends to port 194 (mapped to the CRYPTEXT data bus) the numerical value (between Ø and 255) associated with the I'th character of the string K$ (the key).

The statement OUT 196+I,Ø signals port 196+I (which is between 197 and 2Ø6 inclusive) that the byte currently on the CRYPTEXT data bus should be loaded into the I'th key byte register inside the CRYPTEXT. The Ø in OUT 196+I,Ø is irrelevant and is ignored by the CRYPTEXT; the same effect could be accomplished by OUT 196+I,I.

The statement K$="", which sets K$ to the empty string, reflects a cryptographic principle: unless a key is needed, don't leave it lying around, either in human or machine-readable form!

Line 1Ø4Ø

Line 1Ø4Ø directs CRYPTEXT to begin its initialization sequence (which lasts about 2 milliseconds). Any non-zero value can be used in place of 255; the value does not affect the manner in which CRYPTEXT encodes information.

NOTE: Any attempt to load one or more key-bytes (i.e., whenever an output to one of ports 197 through 2Ø6 occurs) must be followed by an initialization command before encryption is attempted. This is an important security feature of the CRYPTEXT device. Unless initialization is performed, any attempt to load a key-byte will result in "lock-out" of the CRYPTEXT device for encryption or decryption purposes.

Line 1Ø5Ø

Line 1Ø5Ø keeps checking to see if the initialization process is complete: the value of INP(193) represents the byte currently available from the CRYPTEXT's output register. Until initialization is complete, INP(193) will always equal 255. When the initialization process stops, INP(193) will equal the complement of the value supplied to port 195 in line 1Ø4Ø (the complement of 255 happens to be zero).

Repeating the note under line 1Ø4Ø, initialization must follow key-byte entry.

Writing Programs in BASIC

Line 1060

Line 1060 accomplishes the encryption.

One at a time, the bytes representing the characters in the string S$ are sent to port 194 (the CRYPTEXT data bus). The character representing the encrypted byte obtained from port 193 is then appended to the string B$.

PROGRAMMING I/O IN BASIC

Due to the vagaries of BASIC's string manipulation characteristics, it is slightly more difficult to store (perform I/O on) encrypted information on disk or cassette.

The primary element of the problem revolves around "string termination" characters--quotation marks and commas. The encrypted character being stored could be any one of the entire ASCII set, including unprintable characters, quotation marks, commas, or any other. When, for example, an "A" might be encrypted to a quotation mark, and BASIC's string I/O routines would interpret that character as a string terminator, then the string being stored would be terminated, leaving part of the information in limbo.

One way to avoid this problem in BASIC is to convert all information to its numerical form, encrypt that into an array, and store the array. The BASIC routine to do this is short and works well for disk BASIC, but it is very time-consuming to record a numerical file to a cassette.

An alternative method, which allows for fairly rapid storage and retrieval from the cassette, treats all information as string characters.

A program segment to ask for, encrypt, and store information on cassette utilizing strings is as follows:

```
20 INPUT "TEXT ( < 64 CHARACTERS)"; T$
25 IF T$="" THEN U$=CHR$(255): PRINT#-1,U$: END
30 FOR I=1 TO LEN(T$): OUT194,ASC(MID$(T$,I,1))
40 V=INP(193): U$=U$+STR$(V): NEXT I U$=U$+" "
50 CMD"T": PRINT#-1,U$: U$="": T$="": GOTO20
```

A program segment to read the stored information, render it decryptable, and decrypt it is as follows:

```
70 T$="": W$="": CMD"T": INPUT#-1,W$: IF W$=CHR$(255)
   THEN END
80 FOR I=1 TO LEN(W$): U$=MID$(W$,I,1))
90 IF U$<>" " THEN V$=V$+U$: NEXT I
100 W=VAL(V$): V$="": OUT 194,W
110 T$=T$+CHR$(INP(193)): NEXT I
120 PRINT T$: GOTO 70
```

PROGRAM INFORMATION

Line 20 asks for the information to be encrypted. Note that the computer treats the information as a string.

Line 25 sets up a flag to signal the end of the encrypted information.

Line 30 sends each character to the CRYPTEXT to be encoded.

The statement OUT194,ASC(MID$(T$,I,1)), takes the Ith character of T$, converts it to its ASCII numerical form and issues it to the CRYPTEXT.

Line 40 asks the CRYPTEXT for the encrypted character and creates a string of digits which represent the encrypted information.

V=INP(193) returns an ASCII character from the CRYPTEXT. U$=U$+STR$(V) creates and successively appends to U$, each group of digitized characters STR$(V). When the entire line has been encoded, a trailing blank is appended to the string to flag the end of the string during the decryption process.

Line 50 records the string.

CMD"T" disables the interrupts in the Expansion Interface. This command should be deleted when using this program in Level II BASIC.

Line 70 inputs the encrypted information from the cassette recorder, and checks for the flag which was set-up in line 25.

Lines 8∅-1∅∅ return the digitized information to its numerical form, W=VAL(V$), and sends the number to the CRYPTEXT to be decrypted, OUT 194,W.

Line 11∅ re-forms the original line of information as T$.

A complete BASIC program using the above segments together with appropriate key-loading segment appears in this manual in appendix 4.

GETTING TRICKY: CODE BRANCH

You now know how to make CRYPTEXT work for you from BASIC. But CRYPTEXT is versatile enough that it can be used in a variety of ways to build secure information storage and transmission systems. A number of techniques for introducing additional complexity into the encryption process center around the use of CRYPTEXT's unique Code Branch feature.

After initialization is complete, a code branch command may be issued prior to the encryption of any byte. (Before initialization and during key entry, Code Branch has a slightly different effect; see the section of this manual on Security Considerations for more information.)

Code branching is accomplished by signalling port 196 that a branch is desired; executing the BASIC statement OUT 196,∅ performs that task. (Again, the ∅ is irrelevant and is ignored by CRYPTEXT.)

Code branching causes CRYPTEXT's internal code generator to modify itself in a fashion which is unpredictable to the user but which is perfectly repeatable. For example, if a code branch is performed after each 1∅ bytes are encrypted, then a code branch must be performed during decryption after each 1∅ bytes; otherwise subsequent bytes will not decode correctly. Code branches may be performed as often as desired (e.g., before each byte) and are very fast, requiring only a few microseconds to accomplish.

Code branches can also be done at irregular intervals: WARNING: unless the number of bytes between code branches is the same during decryption as during encryption, correct decoding will not occur.

The effect of code branching is also cumulative. Several code branches can be performed together between encryptions; AGAIN, unless the same pattern of code branch is observed during both encryption and decryption, decoding will not be correct.

AND TRICKIER YET

To give a still trickier method, the CRYPTEXT itself can be used in several ways to make the intervals between code branches and/or the number of code branches performed depend indirectly upon the key used. For instance, after initialization one could encrypt a zero and use the resulting pseudorandom number (between Ø and 255) to determine either the interval between subsequent code branches or the number of code branches to be performed at given intervals. Use your imagination; still more complicated schemes are possible.

This is a good place to observe that, in issuing a key to the CRYPTEXT, any character (i.e., any number between Ø and 255) can be issued to any of the key bytes. But not all numbers Ø - 255 represent characters which can be entered via the keyboard. Another way of asking the user for a 1Ø-byte key is to obtain ten elements of an array and to issue these directly to the CRYPTEXT. For example, lines 1Ø2Ø and 1Ø3Ø of our subroutine might be rewritten as follows:

```
1Ø2Ø DEFINT K:  DIM K(9):  FOR I=Ø TO 9:  PRINT
     "ENTER BYTE"; I+1; :  INPUT K(I)
1Ø25 IF(K(I)<Ø) + (K(I)>255) THEN INPUT "RE-ENTER"; K(I)
1Ø26 NEXT I
1Ø3Ø FOR I=1 TO 1Ø:  OUT 194, K(I-1):  OUT196+I, Ø:
     K(I-1) = Ø:  NEXT I
```

Here each of the 1Ø key bytes is entered by the user as a number between Ø and 255 inclusive. In this manner, all the 2^{80} possible keys can be obtained from the keyboard.

THE ICING---SUB-KEYS

In some situations it can be useful to have a secondary or sub-key (sometimes called a working key). Code Branch can be effectively used, following the entry of a master key, to perform the functions required of a sub-key.

In conjunction with EBAC (OUT 194,nn), Code Branch (OUT 196,Ø) can be used to significantly alter the encryption pattern which would normally follow entry of a master key. Here's how it works.

Let us assume you wish to use the character # as a sub-key. The ASCII binary representation of # is: ØØ1ØØØ11. By equating Ø to EBAC, and 1 to CB, our sub-key # would be represented:

```
#  =  Ø = OUT194,Ø
      Ø   OUT194,Ø
      1   OUT196,Ø
      Ø   OUT194,Ø
      Ø   OUT194,Ø
      Ø   OUT194,Ø
      1   OUT196,Ø
      1   OUT196,Ø
```

By issuing these instructions (or any other) to the CRYPTEXT after initialization, its generator will be altered, creating a stream of ciphertext which can only be decrypted by using the master key and the sub-key.

Since the sub-key is completely useless without the master key, less stringent security measures may be appropriate in some circumstances with little fear of compromising the security of encrypted information.

For the use of sub-keys to be optimal, it is important that the same sub-key never be used more than once with any given master key, and, secondly, that it be as long (complex) as practical. (It is doubtful whether sub-keys larger than 60 characters will increase the complexity of the generator)

Writing Programs in BASIC

The following program segment can be appended to the BASIC demonstration program (appendix 4) to allow for the use of a sub-key after master key entry.

```
1050 IF INP(193)=255 THEN 1050
1060 REM  ***   LOAD SUB-KEY   ***
1070 INPUT"ENTER SUB-KEY";S$
1080 FOR I=1 TO LEN(S$): C=128
1090 S=ASC(MID$(S$,I,1))
1100 IF S>=C THEN OUT196,0 ELSE OUT194,0
1110 IF C=1 THEN NEXTI ELSE IF S>=C THEN S=S-C
1120 IF C=1 THEN 1130 ELSE C=C/2: GOTO 1100
1130 RETURN
```

Assembly-language techniques for CRYPTEXT

Programming in assembly language will enable you to take full advantage of the high-speed encryption capabilities of your CRYPTEXT. Figure 1 below provides a listing of a routine to issue a key to the CRYPTEXT and to initialize it; figure 2 provides a routine to encrypt a block of memory locations.

Figure 1:

```
01000 KEYPUT: DI              ;10 KEY BYTES WILL BE ISSUED
01100         LD     B,10
01200         LD     C,197
01300         LD     IX,KEY
01400 KEYB:   LD     A,(IX)
01500         OUT    (194),A ;ISSUE KEY BYTE
01600         CALL   DEL1
01700         OUT    (C),A   ;LOAD & LATCH KEY BYTE
01800         INC    C
01900         INC    IX
02000         DJNZ   KEYB
02100 ZKEY:   LD     HL,KEY
02200         LD     DE,KEY+1
02300         LD     BC,9
02400         LD     (HL),0
02500         LDIR           ;ZERO KEY BUFFER
02600         LD     A,255
02700         OUT    (195),A ;START INITIALIZATION
02800 INITC:  IN     A,(193)
02900         NOP
03000         CP     0       ;INITIALIZATION COMPLETE?
03100         JR     NZ,INITC
03200         EI
03300         RET
03400 DEL1:   PUSH   AF      ;DELAY FOR EBAC
03500         LD     A,1
03600 DEL2:   DEC    A
03700         JR     NZ,DEL2
03800         POP    AF
03900         RET
04000 KEY:    DS     10      ;KEY BUFFER
```

Assembly-language techniques for CRYPTEXT

Figure 2:

```
01000  CRYPT:  LD     HL,BUF1 ;BUF1 HAS 256 BYTES OF TEXT
01100          LD     DE,256
01200  CRYP:   LD     B,32
01300  CRYP1:  LD     A,(HL)
01400          OUT    (194),A ;EBAC
01500          CALL   DEL1
01600          IN     A,(193) ;GEB
01700          NOP
01800          LD     (HL),A  ;SAVE BYTE
01900          INC    HL
02000          DEC    DE
02100          DJNZ   CRYP1
02200          OUT    (196),A ;CODE BRANCH EVERY 32 BYTES
02300          LD     A,D
02400          OR     E
02500          JR     NZ,CRYP
02600          OUT    (196),A ;CB
02700          RET
02800  BUF1:   DS     256     ;TEXT BUFFER
```

Data Security Considerations

Here are miscellaneous suggestions, pointers, rules of thumb, cautionary notes and the like that may help you to use the CRYPTEXT effectively. The applicability of any and all of these must be determined by the user in the light of the specific circumstances and security requirements of each prospective application.

Be aware of the human element. The simplest way for an adversary to inspect information which has been encrypted is to beg, borrow, purchase or steal the plaintext! Careful control over all individuals who have access to plaintext documents is essential, as is the need to either destroy or ensure the physical security of plaintext documents when they are not being used.

Be thoughtful in the choice of keys. Don't use names, telephone or social security numbers, birthdays and the like; they are obvious guesses for anyone attempting to decrypt your information.

Safeguard your keys. Give any key at least the same protection you would accord to a key or lock combination for a vault or safe. If keys must be divulged or recorded, be sure they are available only to authorized individuals. For key distribution, use only controlled channels (such as hand or courier delivery, public-key cryptographic channels or the like). Don't use public channels such as telephone, TELEX, radio, mail and so on unless you can tolerate the potential risks that such channels entail.

Don't use the same key twice. This way, if a key is compromised for any reason, only the plaintext encrypted using that key is compromised. This practice can make a cryptanalyst's job harder, too.

Change keys periodically. From time to time, it is good practice to re-encrypt using a new key unrelated to the current key. This puts time pressure on any adversary who may be able to obtain your key but not your ciphertext. It is doubly important to change keys any time a compromise of security is known or suspected to have occurred, or when there is a change in the list of individuals authorized to have access to keys or to encrypted information.

Data Security Considerations

Paraphrase plaintext which is made public. If decrypted information is disseminated, re-word it slightly or make other minor changes in form and/or content if this is feasible. A cryptanalytic attack is much easier with perfectly matched plaintext and ciphertext than without it. Depending upon the application, there can be a number of other good reasons for this practice as well.

Use CRYPTEXT's Code Branch capability. This was incorporated into the CRYPTEXT design as a security-enhancement feature and to complicate the task of a cryptanalyst or other potential adversary. Code branching can be accomplished at negligible cost in data throughput times even if done frequently. CRYPTEXT Corporation incorporates this feature in proprietary software like the CODEFILE system: Use it in designing your own software too.

The two most important uses of Code Branch (OUT196, Ø): First, to intermittently alter the code stream and second, as a sub-key, are discussed in the section of this manual dealing with BASIC programming techniques. While it is possible to execute CB during key-byte entry and prior to INIT, CRYPTEXT Corporation strongly urges the user to thoroughly experiment before using this technique due to the highly unpredictable manner in which CB acts on the key-bytes before initialization.

CODEFILE

If your TRS-80 is equipped with one or more disk drives operating under Radio Shack's TRSDOS 2.1 operating system, you can easily and rapidly secure <u>any</u> DOS file using CRYPTEXT Corporation's proprietary CODEFILE software (available at additional cost) to drive your CRYPTEXT unit. CRYPTEXT Corporation has incorporated many of the suggestions given in this manual to enable rapid disk file encryption, together with additional features to inhibit "spoofing" of encrypted data intended for storage, or for transmission via telecommunications facilities.

CODEFILE is supplied on a formatted data diskette or, at a slight incremental expense, on a TRSDOS system diskette. (Alternatively, CRYPTEXT Corporation will copy CODEFILE onto a diskette supplied by you, at the lower cost.)

The software is a TRSDOS object file called CODEFILE/CMD and occupies only 2 "granules" (half-tracks) of disk space.

<u>Using CODEFILE</u>:

To use CODEFILE to encrypt a file whose name (filespec) is DATA, for example, is simple and rapid. Suppose you wish to use the key XXXXXXXXXX to encrypt DATA and store the ciphertext in a new file called DATA1. With your CRYPTEXT unit connected and a diskette containing CODEFILE/CMD inserted in your disk drive, you need only type at the keyboard:

```
CODEFILE,DATA,XXXXXXXXXX,DATA1
```

followed by pressing (ENTER). You can then (or later) KILL DATA (kill the data file).

NOTE: The KILL command only prevents logical access to the file, it does not delete or erase the contents of the file. To assure erasure or destruction of the contents of the file, it must be overwritten or magnetically erased. Overwriting the plaintext with the ciphertext version of the file accomplishes destruction of the plaintext, i.e., COPY DATA1 TO DATA.

CODEFILE

To decrypt DATA1, type:

CODEFILE,DATA1,XXXXXXXXXX,DATA2

followed by pressing (ENTER). The file DATA2 will contain the plaintext; i.e., the original contents of DATA.

The files used must have valid TRSDOS filespecs; these may include extensions, drivespecs, or TRSDOS passwords if desired.

Any key can be typed, excepting only that no commas may be used as characters. If fewer than ten key characters are supplied, CODEFILE will pad the key on the right with blanks. If more are supplied, only the first ten will be used. If the first filespec is identical to the second, i.e., you type:

CODEFILE,DATA,XXXXXXXXXX,DATA

the decrypted or encrypted output of the CRYPTEXT will replace the original file, so be careful. Also, as was mentioned in earlier sections, shift A (upper case A) will display on the screen the same as 'a' (lower case a), but shift A results in a different key byte than the lower case a. This can be useful but confusing unless care is used to supply <u>identical</u> keys for encryption and decryption.

Versions of CODEFILE for other operating systems will be supplied on a custom basis. Contact CRYPTEXT Corporation directly for information and price quotations on this and other custom software projects.

CRYPTEXT Command Structure

Command Port	Abbreviation
197-206 (Inclusive) Load Key Byte nn (nn=1,...,10)	LKB nn
193......Get Encrypted Byte	GEB
194......Encrypt Byte/Advance Code	EBAC
195......Initialize	INIT
196......Code Branch	CB

Each command is mapped to its respective port via the microprocessor I/O port address lines.

Commands 194, 195, and no others load the current status of the microprocessor data bus (via strobe) into the CRYPTEXT data buffer.

The 193 (GEB) command is to be used <u>only</u> to obtain input <u>from</u> the CRYPTEXT; all other commands are to be used <u>only</u> to transmit output <u>to</u> the CRYPTEXT.

CRYPTEXT Pin-out Diagram

(TRS-80) Pin Number	Signal Name	CRYPTEXT Function
12	OUT	Peripheral Write Strobe Output
18	D4	Bidirectional Data Bus
19	IN	Peripheral Read Strobe Output
20	D7	Bidirectional Data Bus
22	D2	Bidirectional Data Bus
24	D6	Bidirectional Data Bus
25	AØ	Address Output
26	D3	Bidirectional Data Bus
27	A1	Address Output
28	D5	Bidirectional Data Bus
29	GND	Signal Ground
30	DØ	Bidirectional Data Bus
31	A4	Address Output
32	D2	Bidirectional Data Bus
34	A3	Address Output
35	A5	Address Output
36	A7	Address Output
37	GND	Signal Ground
38	A6	Address Output
39	GND	Un-used on TRS-80 *
40	A2	Address Output

*Alternate power input, diode protected, which accepts +8Vdc to +25Vdc.

CRYPTEXT Test Program

This program can be used to verify that your CRYPTEXT is functioning properly. The output of your CRYPTEXT should match exactly that listed here. If you don't have a line printer, change the LPRINT's to PRINT.

To use this, enter the entire program, attach the CRYPTEXT to your computer and RUN.

CRYPTEXT PRODUCTION TEST

```
10 CLS
20 FORA=1TO10:X=1:FORB=1TO8:GOSUB110
30 OUT194,X:OUT196+A,0
40 GOSUB130
50 X=X*2:PRINT@0,A,B:NEXTB
60 NEXTA
70 FORA=1TO10:FORB=1TO8:GOSUB210
80 NEXTB:LPRINT" ":NEXTA:A=0:B=0
90 GOSUB110   :GOSUB130   :GOSUB210
100 FORX=1TO5:LPRINT" ":NEXTX:END
110 OUT194,0:FORXX=197TO206
120 OUTXX,0:NEXTXX:RETURN
130 OUT195,255
140 IFINP(193)=255THEN140
150 OUT194,0:L(A,B)=INP(193)
160 OUT194,0:M(A,B)=INP(193)
170 OUT196,0
180 OUT194,0:N(A,B)=INP(193)
190 OUT194,0:O(A,B)=INP(193)
200 RETURN
210 LPRINTL(A,B),M(A,B),N(A,B),O(A,B)
220 RETURN
```

CRYPTEXT Test Program

This is the output of the test program.

57	13	35	74
117	254	122	225
8	232	119	188
224	207	8	238
165	57	119	12
200	46	192	43
17	165	217	236
112	73	85	152
187	16	124	107
152	185	234	166
197	83	112	221
88	164	212	254
237	167	83	118
81	39	236	126
170	18	244	238
132	53	239	157
198	20	29	78
1	152	53	130
158	147	60	56
35	95	45	234
41	174	227	53
112	69	153	233
132	118	251	14
163	96	180	58
24	254	194	152
224	166	9	145
94	167	174	117
97	181	71	81
65	237	69	228
79	82	23	162
155	35	188	159
5	151	120	162

CRYPTEXT Test Program

Output continued:

205	150	175	40
88	104	51	39
167	119	84	222
230	40	56	17
21	132	100	61
152	205	238	218
175	36	149	6
100	13	202	50

251	38	31	47
193	243	75	220
162	235	3	71
114	65	111	82
22	131	107	116
60	241	76	64
2	81	71	223
224	243	118	35

149	43	45	25
71	99	105	129
119	176	3	75
28	155	207	238
52	52	157	129
240	112	238	99
110	10	174	40
254	99	231	101

164	179	189	60
228	125	182	127
120	40	201	43
161	162	239	218
130	167	55	94
24	244	153	194
75	30	177	74
203	195	51	105

CRYPTEXT Test Program

Output continued:

255	174	37	52
231	165	97	177
167	149	182	135
161	0	210	221
87	151	7	85
31	35	173	152
53	88	111	46
70	45	205	84
177	117	53	225
67	62	101	120
126	221	203	111
235	144	10	153
122	151	80	61
157	149	199	231
250	65	150	181
47	211	224	91
11	139	123	41

BASIC Demonstration Program

```
10 CLEAR 1000: GOSUB 1010
20 INPUT"TEXT ( < 64 CHARACTERS)";T$
25 IF T$="" THEN U$ =CHR$(255): PRINT#-1,U$: END
30 FOR I=1 TO LEN(T$): OUT 194,ASC(MID$(T$,I,1))
40 V=INP(193): U$=U$+STR$(V): NEXT I: U$=U$+" "
50 CMD"T": PRINT#-1,U$: U$="": T$="" : GOTO 20
60 CLEAR 1000: GOSUB 1010
70 T$="": W$="": CMD"T": INPUT#-1,W$: IF W$=CHR$(255) THEN END
80 FOR I=1 TO LEN(W$): U$=MID$(W$,I,1)
90 IF U$<>" " THEN V$=V$+U$: NEXT I
100 W=VAL(V$): V$="": OUT 194,W
110 T$=T$+CHR$(INP(193)): NEXT I
120 PRINT T$: GOTO 70
1010 CLS: DEFINT A-Z: INPUT"ENTER KEY ( 10 CHARACTERS )";K$
1020 IF LEN(K$)<>10 THEN 1010 ELSE FOR I=1 TO 10
1030 OUT 194,ASC(MID$(K$,I,1)): OUT 196+I,0
1040 NEXT I: OUT 195,255
1050 IF INP(193)=255 THEN 1050 ELSE RETURN
```

RUN this program to encrypt information one line at a time. To stop execution at the end of the information type (ENTER) at the start of a new line.

RUN 60 to decrypt previously encrypted/stored information.

NOTE: CMD"T" must not be used if an Expansion Interface is not connected to your computer.

GLOSSARY

Ciphertext	A message or text which is meant to be understood only by those who have the key or code to it.
Code Branch CB	A CRYPTEXT command (OUT 196,Ø) which changes the pseudo-random generator inside the CRYPTEXT unit in an unpredictable manner.
CODEFILE	A machine language program which uses a CRYPTEXT to encrypt a file, or to decrypt a previously encrypted file.
Cryptanalysis	The process of systematically applying methods and techniques to defeat or "break" a cryptographic system or device.
Decrypt	To render an encrypted message readable.
Encrypt Byte Advance Code EBAC	A CRYPTEXT command (OUT 194,nn) that loads a character (from Ø to 255) into the CRYPTEXT and then advances the CRYPTEXT generator to the following code state.
Encrypt	To hide or make something secret. To render information meaningless to all except those who are privy to a special key or code.
Get Encrypted Byte GEB	A CRYPTEXT command (INP(193)) that loads a character (Ø to 255) from the CRYPTEXT into the computer.
Initialize INIT	A CRYPTEXT command (OUT 195,nn) that prepares the CRYPTEXT generators to begin the process of secure encryption.
Key	Secret information (usually letters or numbers) which allow the holder of the key and the message to decypher the message. The key unlocks the access to the information.
Plaintext	Information in human- or machine-readable form.

GLOSSARY

Public Key	A cryptographic system in which the decryption key is different from the encryption key so that the encryption key can be made public.
Spoofing	Tampering with part of an encrypted information stream without altering the entire text. Spoofing can occur as a result of a system failure or as a result of deliberate attempts to alter the information stream.
Spoof-Proof	A characteristic of a cryptographic system which causes the loss of large amounts of information when very small portions of the information are altered or tampered with. A spoof-proof system allows for relatively easy detection of spoofing problems.